Execução e Estratégias para Mitigação de Ataques de Negação de Serviço em Servidores Web: Um Relato de Experiência

Ricardo de la Rocha Ladeira[1], Ricardo Borges Almeida[2]

[1]Instituto Federal Catarinense, Campus Blumenau – Brasil

[2]Universidade Federal de Pelotas, Campus Porto – Brasil

ricardo.ladeira@ifc.edu.br, rbalmeida@inf.ufpel.edu.br

Abstract. *This paper describes the execution of denial of service attacks performed on web servers configured with different Apache versions, trying to reflect scenarios commonly found in Federal Institutions of Higher Education in Brazil. The approach consisted of the following steps: (i) tool selection for the experiment; (ii) definition of attack parameters; (iii) execution of attack with a selected tool in (i) on servers with similar characteristics to those in production; and (iv) execution of an attack with changes in the firewall as a strategy to mitigate the attack. Based on the observed result, the default configuration does not protect servers from simple denial of service attacks, but the settings implemented in (iv) were sufficient to prevent similar attacks from being successful in future executions.*

Keywords: *Denial of Service, Computer Security.*

Resumo. *Este artigo descreve a execução de ataques de negação de serviço realizados em servidores web configurados com diferentes versões do Apache, buscando refletir cenários comumente encontrados em Instituições Federais de Ensino Superior do Brasil. A abordagem consistiu nas seguintes etapas: (i) seleção de ferramenta para o experimento; (ii) definição de parâmetros de ataque; (iii) execução de ataque com uma ferramenta selecionada em (i), em servidores com características semelhantes aos em produção; e (iv) execução de ataque com alterações no firewall como estratégia para mitigação do ataque. Com base no resultado observado, a configuração padrão não protege os servidores de ataques de negação de serviço simples, mas as configurações implementadas em (iv) foram suficientes para impedir que ataques semelhantes aos que foram realizados obtenham sucesso em execuções futuras.*

Palavras-chave: *Negação de Serviço. Segurança Computacional.*

1. Introdução

No contexto atual, a quantidade de ameaças digitais existentes é expressiva e tem preocupado a comunidade. O cenário brasileiro, em especial, apresenta dados que sustentam esta afirmação. Segundo o [CERT.br 2018], os últimos anos foram aqueles em que ocorreu maior número de notificações de incidentes de segurança.

Entre os tipos de ataques reportados ao CERT.br em 2017, os mais presentes foram os de varreduras em redes (também chamados de *scans*). Em segundo lugar ficaram osataques de negação de serviço (também chamados de Denial of Service — DoS), como mostra a Figura

1, extraída do [CERT.br 2018]. Segundo a [Nexusguard 2017], empresa especializada em mitigação de ataques de negação de serviço, no primeiro quadrimestre de 2017 houve um crescimento de 380% de ataques desse tipo.

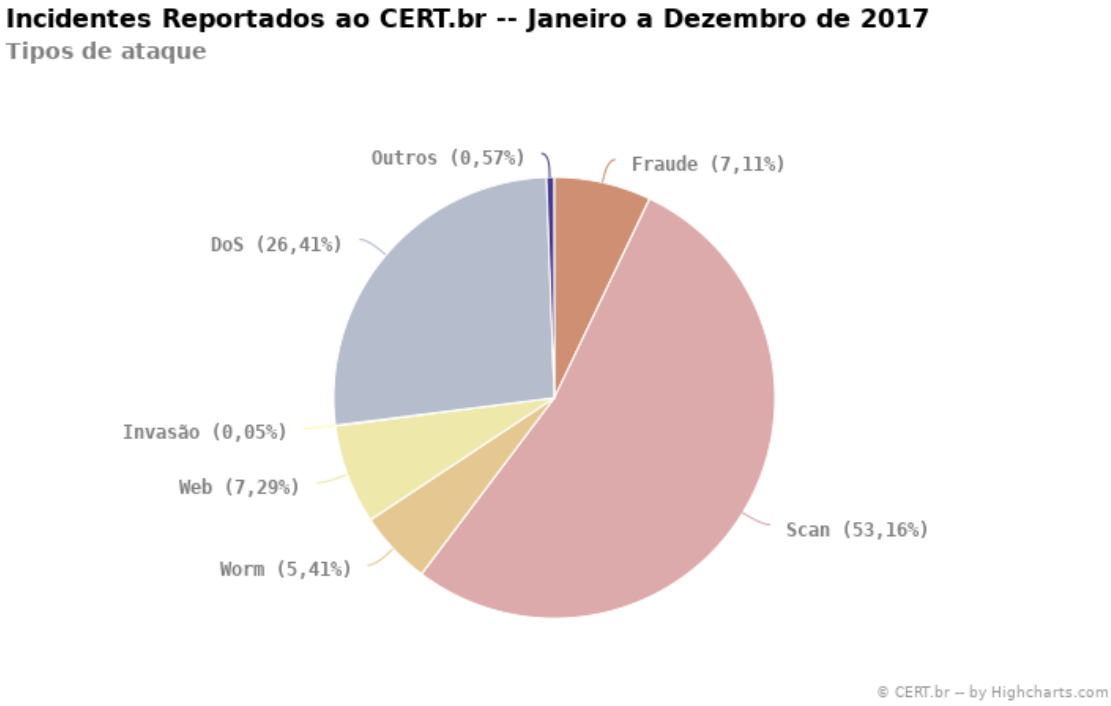

Figura 1. Tipos de ataques reportados ao CERT.br.

Fonte: CERT.br, 2018.

Neste sentido, o crescimento destes ataques aliado às características das infraestruturas de Tecnologia da Informação — TI das Instituições Federais de Ensino Superior do Brasil, que, em especial, necessitam e utilizam aplicações *web* para disseminação de informações, determinaram o foco deste trabalho: o relato de um experimento controlado de execução e mitigação de ataques de negação de serviço em servidores *web*.

Este trabalho descreve o projeto e a execução de um experimento de ataques de negação de serviço utilizando uma ferramenta de teste de estresse para servidores Apache com configurações padrão e de diferentes versões, sem preocupação com questões de *hardening*[1] ou contramedidas específicas de combate a estes ataques. Esta escolha se deu pela experiência dos autores, a qual indica que a "instalação padrão" é um comportamento usual em equipes de TI, a despeito de sua suscetibilidade a ataques [Omar 2017]. Além disso, ressalta-se a necessidade de elaboração e cumprimento de PSI (Políticas de Segurança da Informação) institucionais, o que nem sempre ocorre. A proposta, detalhada no Capítulo 3, consistiu resumidamente em escolher uma ferramenta para testar o servidor e definir o volume de requisições na tentativa de forçá-lo a ficar indisponível ou demonstrar suas ações para situações contingenciais. Na sequência, aplicaram-se novos controles e o ataque foi refeito visando à mitigação da ocorrência destes.

1 Para [Wang et al. 2014], *hardening* consiste em um conjunto de técnicas de fortalecimento que melhoram a segurança das redes ou dos dispositivos de redes existentes através de soluções de segurança ou alterações de configuração.

O presente artigo está organizado em seis capítulos, incluindo esta introdução. O Capítulo 2 traz uma visão geral sobre o ataque de negação de serviço. No Capítulo 3 está detalhado o projeto de experimento, listando as ferramentas utilizadas e explicando o funcionamento da ferramenta utilizada para realização do ataque, da arquitetura alvo e a execução do ataque. O Capítulo 4 discute os resultados e o Capítulo 5 aborda as considerações finais do artigo e o Capítulo 6 traz as perspectivas de continuidade da pesquisa.

2. Negação de Serviço

Negação de serviço é um tipo de ataque realizado para tornar um serviço indisponível. Entre as três principais propriedades de Segurança da Informação — integridade, disponibilidade e confidencialidade —, ele tem como objetivo afetar majoritariamente a disponibilidade do serviço atacado. [Lau et al. 2000] complementam, afirmando que

> "Uma negação de serviço é caracterizada por uma tentativa explícita de um invasor de impedir que usuários legítimos usem recursos. Um invasor pode tentar: "inundar" uma rede e, assim, reduzir a largura de banda de um usuário legítimo, impedir o acesso a um serviço ou interromper o serviço para um sistema ou usuário específico" (tradução nossa).

Estes ataques podem ser realizados, por exemplo, por usuários maliciosos que obtêm acesso a um serviço se aproveitando de uma senha fraca ou do uso de algum recurso que possua uma vulnerabilidade conhecida, tal como um software desatualizado. Assim, por exemplo, é possível parar a execução de um serviço.

No entanto, uma forma comum de afetar a disponibilidade de um serviço é consumindo seus recursos essenciais, tais como o uso de memória, o processamento, o espaço em disco e a largura de banda [Laufer et al. 2005]. Ao enviar requisições para um servidor e consumir boa parte de seus recursos, realiza-se o que se chama de "ataque por inundação" ou flood. Quando o ataque ocorre na camada de aplicação, diz-se que o ataque é um ADoS (*Application Denial of Service*).

Apesar da Internet das Coisas e suas vulnerabilidades estarem contribuindo para a realização de ataques distribuídos de negação de serviço (ou *Distributed Denial of Service*, também conhecido pelo acrônimo DDoS) baseados em volume, a [Nexusguard 2017] também identificou que os ataques na camada de aplicação predominaram, superando os ataques volumétricos realizados em camadas inferiores na pilha TCP/IP (*Transmission Control Protocol/Internet Protocol*).

A elaboração de contramedidas nem sempre é trivial, já que é difícil identificar as requisições maliciosas em meio ao tráfego legítimo em ataques de inundação. Isto acontece porque as requisições ilegítimas se assemelham às legítimas [Laufer et al. 2005, Ranjan et al. 2006]. O ataque de negação de serviço por inundação *Hypertext Transfer Protocol* — HTTP-GET é um exemplo disso.

Além disso, os atacantes estão conscientes de que o tráfego HTTP não é bloqueado por equipamentos de segurança por padrão [Singh et al. 2017], sendo necessárias ações dos administradores dos serviços. Entre estas ações, há uma proposta de método de defesa contra ataques na camada de aplicação com estratégias seletivas [Dantas et al. 2014], revisões da literatura que identificam trabalhos que detectam ataques DDoS do tipo SlowHTTP [Tripathi et al. 2016] e propostas de um sistema anti-inundação HTTP-GET de ataques DDoS [Mirvaziri 2017].

Neste trabalho, pretende-se verificar a possibilidade de mitigação de ataques de negação de serviço através de inundação por requisições HTTP-GET[2] sem a necessidade de instalação de ferramentas externas, mas com uma simples alteração na política de *firewall*, voltando-se aos ataques simples (não distribuídos). O experimento está descrito no capítulo subsequente.

3. Métodos

O experimento projetado foi dividido em quatro etapas listadas a seguir e descritas na sequência deste capítulo:

1. Seleção do conjunto de ferramentas a serem utilizadas no experimento;
2. Definição de parâmetros de ataque;
3. Execução do primeiro ataque, com a ferramenta selecionada, em servidores Apache instalados seguindo as configurações padrão, ou seja, com características semelhantes aos geralmente identificados em ambientes de produção;
4. Execução do segundo ataque, em condições iguais às do primeiro ataque, porém com alteração de configuração no *firewall*.

Conforme apontam [Guan et al. 2014], são inúmeras as ferramentas que tornam possível a realização de testes de desempenho de servidores *web*. Uma das alternativas gratuitas neste sentido, e escolhida neste trabalho, é o siege, um aplicativo de linha de comando que permite simular cenários em que múltiplos usuários acessam concorrentemente um recurso em uma solução *web*. Maiores detalhes sobre o siege, incluindo medidas de desempenho e alguns dos parâmetros utilizados serão discutidos na subseção seguinte.

Para monitorar o progresso do ataque em termos de consumo de recursos, optouse por utilizar htop. Conforme [Muhammad 2018], o htop é uma ferramenta de linha de comando para sistemas *Unix-like* capaz de exibir os processos em execução dinamicamente em tempo real.

Uma ferramenta importante e presente na proteção da infraestrutura de serviços da instituição atacada nos testes é o *firewall*. Segundo [Batistela and Trentin 2009], eles são capazes de bloquear acessos indevidos aos serviços, sendo uma ferramenta importante para o combate ao comprometimento das propriedades[3] de Segurança da Informação. O *firewall* escolhido para o segundo ataque foi o iptables[4] versão 1.4.21, pois além de ser sem custo, ele é fornecido na maioria das distribuições Linux e é amplamente adotado pelos administradores de rede. Na instituição em questão, o iptables já era utilizado antes do experimento. No entanto, entre as regras já aplicadas antes da realização do ataque, as únicas relacionadas ao experimento eram as de aceitação de pacotes pelas portas 80, relativa a pacotes HTTP, e 443, relativa a pacotes HTTPS.

Para realizar o ataque, definiu-se que o tempo de cada teste não excederia duas horas, e que, para cada versão do Apache, o ataque seria realizado somente uma vez. Este tempo foi escolhido por ser considerado, pelos autores, suficiente para verificar se a carga de requisições

---

2 Os experimentos deste trabalho limitam-se a ataques por requisições HTTP-GET, utilizando, portanto, a arqutetura REST (*Representational State Transfer*). O artigo não cobre a arquitetura SOAP por esta não ser utilizada nas instituições nas quais os testes foram realizados.

3 Integridade, disponibilidade e confidencialidade.

4 https://www.netfilter.org/projects/iptables/index.html

poderia tornar o servidor atacado indisponível. Em testes preliminares, todas as requisições concorrentes foram geradas em intervalo inferior a cinco minutos. Assim, além da criação das requisições concorrentes, seria possível observar neste tempo se o servidor vítima possuiria eventuais contramedidas implementadas, observando assim o seu comportamento durante o ataque.

Definiu-se ainda que o ataque seria realizado por somente um computador. O ataque distribuído será efetuado em trabalhos futuros, na mesma instituição na qual os ataques foram originados, pois esta dispõe de recursos para elaborar o experimento.

A expectativa era de que o resultado mais provável fosse o atingimento de alto consumo de CPU no servidor e oscilação[5] no tempo de resposta para usuários legítimos, o que poderia ser verificado, entre outras formas, realizando uma requisição legítima ao servidor ou aguardando respostas a requisições de pacotes ICMP (*Internet Control Message Protocol*), utilizando a ferramenta ping, por exemplo. Esta expectativa baseia-se em [Laufer et al. 2005], que afirmam que muitas vezes o tráfego gerado por apenas uma estação de ataque não é suficiente para exaurir os recursos da vítima, e que a indisponibilidade é mais possível de acontecer quando a vítima dispõe de poucos recursos. Um segundo possível resultado seria a total indisponibilidade do servidor.

Embora o ataque não fosse distribuído, a proposta era simular usuários concorrentes. A quantidade definida para usuários concorrentes foi 640, e o tempo definido entre as requisições foi de um segundo por ser o mínimo possível, tentando assim inundar/indisponibilizar o sistema. O número de usuários concorrentes foi escolhido em função das limitações da rede de origem do tráfego e, além disso, de acordo com [Fulmer 2012], é considerado um número alto, com base em informações disponíveis no FAQ da ferramenta. Ainda com base no FAQ, há relatos de casos de problemas com cerca de 800 usuários concorrentes simulados [Fulmer 2012]. Portanto, optou-se por um número inferior para diminuir as chances de problemas com a ferramenta.

O objetivo proposto para o primeiro ataque era verificar se os controles ativos a partir de instalações padrão de diferentes versões do servidor Apache reagiriam satisfatoriamente às múltiplas requisições criadas. Do ponto de vista do atacante, o objetivo era realizar requisições unicamente para tornar o servidor *web* indisponível.

Após esta ação, uma proposta de configuração no *firewall* com definição do *rate limit* foi implementada. O *rate limit* (taxa limite, em tradução livre) define um valor limiar para cada IP que realiza conexão com o servidor. O valor utilizado no *rate limit* foi 5 neste experimento, o que significa que, pra cada IP, cinco conexões simultâneas por minuto poderiam existir. Esta regra foi implementada na terceira etapa, que consistiu no segundo ataque. O principal resultado esperado era de efetividade dos controles implementados, tornando impossível o comprometimento da disponibilidade do servidor. Um segundo resultado possível seria a oscilação no tempo de resposta.

É necessário ponderar que há ameaças à validade dos resultados do experimento. Entre as possíveis ameaças identificadas estão eventuais oscilações de largura de banda e problemas em equipamentos físicos, comportamentos atípicos nas ferramentas selecionadas e eventuais problemas no sistema operacional, em processos em execução ou no consumo de recursos nas máquinas atacante e vítima, não oriundos do experimento.

5 Também conhecida como *jitter*.

3.1. Ferramenta siege

Segundo [Fulmer et al. 2018], Siege[6] (siege, na linha de comando para sistemas *Unix-like*) é uma ferramenta utilitária que realiza testes de estresse (ou testes de carga HTTP) e *benchmarking* em servidores *web*. Essa ferramenta permite que um número parametrizável de usuários simultâneos sejam simulados para requisições concomitantes, permitindo identificar os limites do servidor. Desta forma, a ferramenta pode ser usada para fins de auditoria e medição, por usuários internos. No entanto, o siege também pode servir ao público externo possibilitando a execução de atividades maliciosas, especialmente àqueles que procuram afetar deliberadamente a disponibilidade de serviços executados pelo servidor.

As medidas de desempenho incluem o tempo decorrido durante a execução do teste, a quantidade de dados transferidos (incluindo os cabeçalhos das requisições), o tempo de resposta do servidor avaliado, o número de transações por segundo, o volume de dados transferidos por segundo, o número de usuários simultâneos em um período específico de tempo e o número de vezes que retornou "OK". A ferramenta tem essencialmente três modos de operação: regressão, simulação de internet e força bruta. Segundo [Fulmer et al. 2018], o modo regressão consiste em ler um grande número de URLs (*Uniform Resource Locators*) de um arquivo de configuração (/etc/urls.txt) e executálas de forma incremental. O modo de simulação de internet é idêntico ao de regressão, mas a execução é aleatória. O modo força bruta consiste na seleção de uma única URL com uma configuração de tempo de execução na linha de comando.

Entre os parâmetros da ferramenta, o usuário pode definir a quantidade de usuários concorrentes simulados utilizando -c (o padrão é 10), o intervalo de tempo entre a requisição de cada usuário, utilizando -d, e o tempo de duração do experimento utilizando -t.

3.2. Arquitetura Alvo

Para realização dos testes, o ambiente de avaliação consistiu em uma máquina virtual com 6 *Central Processing Units* — CPUs Intel(R) Xeon(R) CPU E5-2660 com frequência de 2.20GHz, cada uma com dois núcleos virtuais, cache de 20480 KiB e 15,7 GiB de memória *Random Access Memory* — RAM. A máquina executava uma distribuição Debian 8 de 64 bits. Esta máquina continha diferentes versões do Apache instaladas com suas configurações padrão mantidas, e consistia de um clone da máquina utilizada para hospedar o portal da instituição. A proposta de utilizar um clone com as mesmas configurações advém da necessidade de simular o ataque contra uma máquina semelhante à original, porém sem afetar os usuários legítimos.

Para determinar as versões do Apache, foi utilizado o Ansible[7] para coletar as versões existentes no ambiente de produção na instituição na qual os testes foram realizados. Com isso, as seguintes versões foram testadas: 2.2.14, 2.2.16, 2.2.22, 2.4.7 e 2.4.10. Para algumas dessas versões, a fim de facilitar o processo, foi criado um ambiente de testes, sendo realizada a clonagem de máquinas virtuais existentes, mantendo as aplicações *web* conforme disponíveis no ambiente de produção. Esta ação permite realizar os ataques em um ambiente controlado, sem ocasionar transtornos aos usuários legítimos. A Figura 2 ilustra graficamente o ataque considerando as máquinas atacante e atacada.

---

6 https://www.joedog.org/siege-home/

7 Ansible é uma ferramenta que automatiza o gerenciamento de múltiplas máquinas no sentido de implantar sistemas e aplicações e consultar informações.

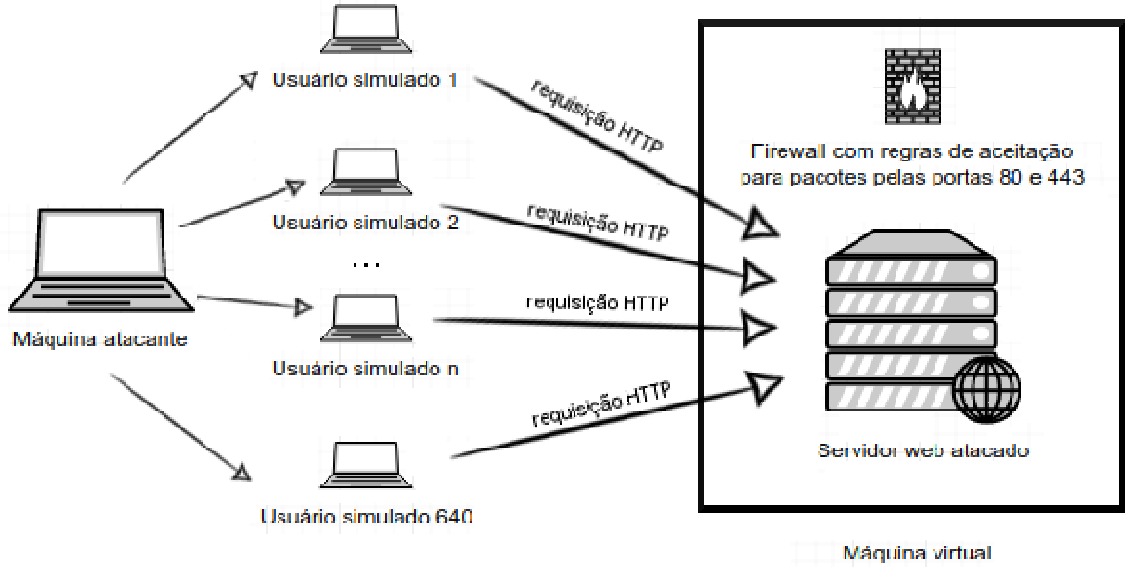

Figura 2. Ilustração gráfica do primeiro ataque projetado considerando a arquitetura alvo. Fonte: **elaborado pelos autores, 2018.**

Fonte: elaborado pelos autores, 2018.

3.3. Execução do Ataque

O ataque contou com a colaboração de dois pesquisadores e foi realizado em março de 2017. Na ação, um pesquisador, externo à instituição, realizou o ataque utilizando a ferramenta siege, em uma máquina com 7,7 GiB de memória RAM e um processador Intel(R) Core i3(R) CPU 2310M com frequência de 2.10 GHz e quatro núcleos virtuais, com sistema operacional Ubuntu 14.04 LTS de 64 *bits*.

O segundo pesquisador, interno à instituição atacada, foi responsável por realizar a instalação e a clonagem de máquinas e acompanhar o desempenho do servidor por meio da ferramenta htop, verificando o consumo da máquina e monitorando a disponibilidade do servidor. Esta ação pode ser observada através da captura de tela apresentada na Figura 3. Nela é possível ver diversos registros de www-data na coluna user, que representam os usuários simulados pelo siege pelo lado do servidor .

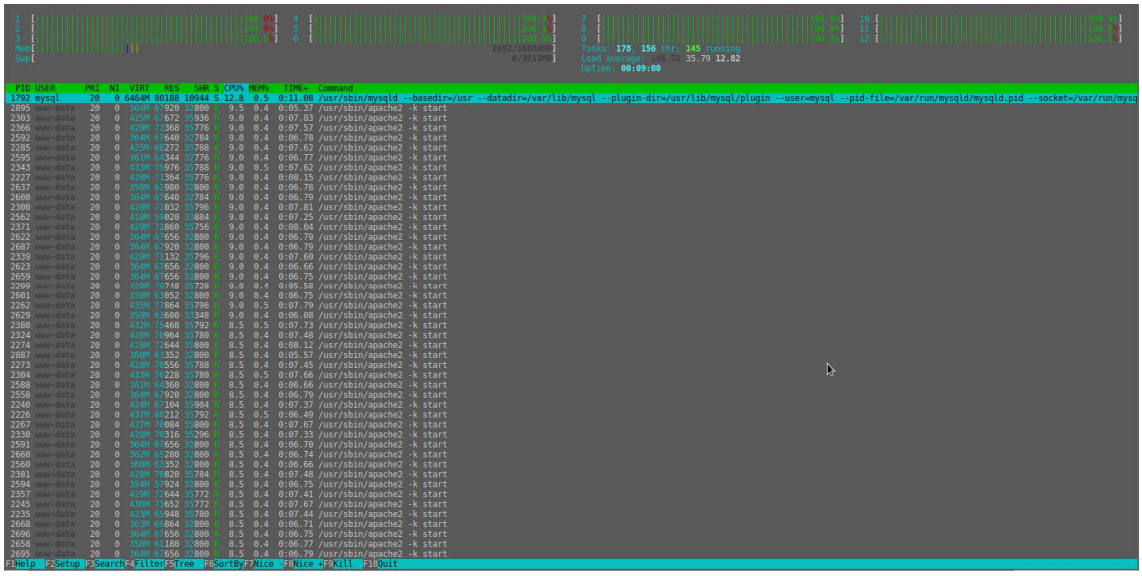

Figura 3. Monitoramento de um servidor atacado através da ferramenta htop.
Fonte: elaborado pelos autores, 2017.

Para cada versão do Apache o teste foi realizado uma vez. Após cerca de cinco minutos, verificou-se a indisponibilidade do sistema e a execução do siege foi encerrada antes do tempo previsto. Nos testes realizados, todas as versões avaliadas do Apache se mostraram vulneráveis ao ataque, ou seja, os servidores ficaram indisponíveis.

Após o primeiro ataque, os servidores Apache foram reconfigurados da mesma forma, mas o *firewall* instalado (iptables) foi configurado com o valor 5 (cinco) atribuído ao *rate limit*. Desta vez, o *firewall* permitiu cinco conexões por IP e bloqueou todas as tentativas a partir da sexta conexão concorrente, assim não afetando a disponibilidade do servidor *web* para a segunda tentativa de ataque de negação de serviço. Não houve grande oscilação entre o tempo de resposta de requisições realizadas com a ferramenta ping, um resultado esperado.

Todos os ataques foram realizados sob os parâmetros definidos no projeto do experimento, declarados no início deste capítulo. O comando utilizado na execução dos testes foi o seguinte:

siege -c640 -t2h -d1 endereco.do.portal.

O parâmetro -c640 indica que foram utilizados 640 usuários concorrentes simulados. O parâmetro -t2h representa que o tempo de duração do ataque foi definido para duas horas. O parámetro -d1 define que o intervalo de tempo entre a requisição de cada usuário foi de um segundo. O endereco.do.portal representa o endereço a ser atacado. No caso dos experimentos realizados, o ataque poderia ser realizado diretamente através do IP da vítima.

Há duas observações importantes a serem destacadas: para os casos em que a máquina original foi clonada, para realizar o ataque pelo domínio original da máquina clonada, a definição do IP foi realizada manualmente no arquivo /etc/hosts da máquina em que o siege foi executado. A segunda é que a ação de definição do *rate limit* prospera somente para ataques simples ou de pouca distribuição. Para citar um exemplo, ao utilizar o mesmo ataque concorrente de 128 *hosts* distintos, com o *rate limit* permitindo somente 5 requisições simultâneas, atinge-se os mesmos 640 usuários simultâneos que no primeiro ataque e, assim, o ataque pode obter o mesmo número de requisições e indisponibilizar novamente o servidor atacado.

Sugere-se ainda que, além do *rate limit*, sejam definidos controles também no servidor *web*. Utilizando como exemplo o servidor Apache, utilizado nos testes, é possível definir a quantidade de clientes que podem ser atentidos através do atributo MaxClients, definido no arquivo de configuração do servidor (normalmente disponível em /etc/apache2/apache2.conf). Este parâmetro contribui como um obstáculo também para ataques distribuídos de negação de serviço. Outras opções de segurança para o servidor Apache, incluindo parâmteros para contenção de DoS, podem ser encontradas em https://httpd.apache.org/docs/trunk/misc

/security_tips.html.

Testes também foram realizados com o modsecurity, utilizando as regras experimentais da *Open Web Application Security Project* (Projeto Aberto de Segurança em Aplicações *Web*) — OWASP[8] versão 2.2.9, habilitando modsecurity_crs_11_dos_protection.conf e modsecurity_crs_11_slow_dos_protection.conf. É importante destacar que, em especial, para proteção de *slow-dos*, as regras utilizam o RequestReadTimeout, o qual deve ser ajustado de acordo com o cenário. Nos testes realizados, foi possível identificar falsos positivos especificamente ao realizar *upload* de arquivos para a aplicação *web*.

4. Discussão dos Resultados

O primeiro teste demonstrou a falta de preparo dos servidores Apache, em configuração padrão, para lidar com ataques de negação de serviço, sem haver também controles efetivos de outras ferramentas. Em cerca de cinco minutos, uma única máquina conectada através da internet com 640 usuários simultâneos simulados foi capaz de deixar os servidores indisponíveis.

Os servidores avaliados, após reconfiguração do *firewall*, atenderam às expectativas e eliminaram sumariamente as possibilidades de ataque de negação de serviço simples. No entanto, ressalta-se que o conjunto de configurações sugeridas ainda está em fase de implantação. Deseja-se realizar novos testes em um novo ambiente para comparação dos resultados e definição de novas métricas, bem como de novos tipos de ataques de DoS.

Além disso, é importante ponderar que o aumento no número de usuários e de acessos a determinados servidores *web* na instituição é uma preocupação central, uma vez que a indisponibilidade ou lentidão na operação de um site ou serviço pode resultar em consequências negativas, tanto para quem dele depende quanto para a instituição.

O siege, portanto, serviu também como um recurso para realização de testes de *benchmarking*, permitindo futuros ajustes de parâmetros para melhorar o desempenho dos servidores Apache, identificando seus limites. Mesmo com as configurações padrão mantidas em todas as versões testadas, a despeito da necessidade de estudo e reconfiguração destas com base em boas práticas, o suporte do *firewall* já demonstrou um ganho à instituição considerando o cenário real.

Com base nos resultados já observados neste experimento, a equipe de suporte da instituição onde os testes foram realizados foi capaz de melhorar a disponibilidade e minimizar as possibilidades de ataques em seus servidores.

8 https://github.com/SpiderLabs/owasp-modsecurity-crs/tree/v2.2/master

5. Considerações Finais

Neste trabalho, apresentou-se um experimento de ataque de negação de serviço a servidores *web* instalados seguindo as configurações padrão do Apache. O experimento representou uma ação importante para a equipe de tecnologia da instituição na qual os testes foram realizados, haja vista que ataques semelhantes podem ser desferidos por usuários maliciosos.

Embora o siege seja um simples utilitário de linha de comando e tenha sido utilizado neste trabalho com o objetivo principal de avaliar questões relacionadas a DoS, o aplicativo também foi útil para testes de avaliação de desempenho de servidores *web* em situações de uso crescente.

6. Trabalhos Futuros

A ação realizada foi importante no sentido de testar novas configurações e impedir ataques futuros. Porém, faz-se necessário ainda testar novos cenários de configuração e mensurar os resultados para identificar os melhores parâmetros para reagir a ataques do tipo negação de serviço. Pretende-se, ainda, expandir o rol de servidores e instituições envolvidas.

Deseja-se projetar cenários distintos para a continuidade da pesquisa, aplicando configurações diferentes no *firewall*, nos servidores *web*, e implementando contramedidas como monitoramento de pacotes e espalhamento, a fim de identificar outras vulnerabilidades. Além disso, pretende-se realizar novos testes com ataques que envolvam não apenas métodos HTTP-GET, mas também POST.

Planeja-se também a realização de ataques coordenadamente distribuídos, verificando a eficácia dos cenários configurados em mitigar ataques de negação de serviço distribuídos (DDoS). Outras propostas envolvem a elaboração de um framework, explorando boas práticas estabelecidas na PSI das instituições envolvidas, e uma cartilha de boas práticas para configuração de servidores voltada à mitigação de ataques.

Agradecimentos

Os autores agradecem à Universidade Federal de Pelotas e ao Instituto Federal Catarinense, que tornaram possível a realização deste trabalho.

Referências

Batistela, V. and Trentin, M. A. (2009). Identificação e análise de tráfego malicioso através do uso de honeypots. *Revista Brasileira de Computação Aplicada*, 1(1):2–14.

CERT.br (2018). Estatísticas dos incidentes reportados ao cert.br. Disponível em: http://www.cert.br/stats/incidentes/. Acesso em: 14 abr. 2018.

Dantas, Y. G., Nigam, V., and Fonseca, I. E. (2014). A selective defense for application layer ddos attacks. In *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, pages 75–82. IEEE.

Fulmer, J. (2012). Siege frequently asked questions. Disponível em: https://www.joedog.org/siege-faq/. Acesso em: 20 jul. 2018.

Fulmer, J. et al. (2018). *siege*. General Commands Manual.

Guan, X., Cheng, B., Song, A., and Wu, H. (2014). Modeling users' behavior for testing the performance of a web map tile service. *Transactions in GIS*, 18:109–125.

Lau, F., Rubin, S. H., Smith, M. H., and Trajkovic, L. (2000). Distributed denial of service attacks. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pages 2275–2280. IEEE.

Laufer, R. P., Moraes, I. M., Velloso, P. B., Bicudo, M. D., Campista, M. E. M., Cunha, D. d. O., Costa, L., and Duarte, O. (2005). Negação de serviço: Ataques e contramedidas. *Livro Texto dos Mini-cursos do V Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Disponível em: http://professor.ufabc.edu.br/~joao.kleinschmidt/aulas/seg2013/negacao.pdf. Acesso em: 24 jul. 2018.

Mirvaziri, H. (2017). A new method to reduce the effects of http-get flood attack. *Future Computing and Informatics Journal*, 2(2):87–93.

Muhammad, H. (2018). *htop(1) Linux man page*. Página do manual escrita por Bartosz Fenski.

Nexusguard (2017). Distributed denial of service (ddos) threat report q1 2017. Disponível em: https://cdn2.hubspot.net/hubfs/466726/Q1_2017_Threat_Report/Nexusguard_DDoS_Threat_Report_Q1_2017_EN.pdf. Acesso em: 14 abr.2018.

Omar, S. (2017). *Information system security threats and vulnerabilities: evaluating the human factor in data protection*. PhD thesis.

Ranjan, S., Swaminathan, R., Uysal, M., and Knightly, E. W. (2006). Ddos-resilient scheduling to counter application layer attacks under imperfect detection. In *INFOCOM*. Citeseer.

Singh, K., Singh, P., and Kumar, K. (2017). Application layer http-get flood ddos attacks: Research landscape and challenges. *Computers & security*, 65:344–372.

Tripathi, N., Hubballi, N., and Singh, Y. (2016). How secure are web servers? an empirical study of slow http dos attacks and detection. In *Availability, Reliability and Security (ARES), 2016 11th International Conference on*, pages 454–463. IEEE.

Wang, L., Jajodia, S., Singhal, A., Cheng, P., and Noel, S. (2014). k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 11(1):30–44.